ISSN 1410 - 9158

Majalah Ilmiah
Informatika, Komputer, dan Bisnis

FORMAT

Volume 8, Nomor 2, Mei 2007

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER AKAKOM
YOGYAKARTA

Majalah Ilmiah
Informatika, Komputer, dan Bisnis

FORMAT

PELINDUNG:
Ketua Yayasan Pendidikan Widya Bakti

KETUA UMUM:
Ketua STMIK AKAKOM Yogyakarta

KETUA DEWAN REDAKSI:
Bambang P. D. P., S.E., Akt., S.Kom., M.MSi.

ANGGOTA DEWAN REDAKSI:
Ir. F. Soesianto, B.Sc.E., Ph.D.
Prof. H. Adhi Susanto, M.Sc., Ph.D.
Drs. Tri Prabawa, M.Kom.
Ir. Surjono, M.Phil.
Ir. Sudarmanto, M.T.
Ir. M. Guntara, M.T.
Ir. Totok Suprawoto, M.M.
Budi Sugihardjo, S.E., M.M.
Heru Agus Triyanto, S.E., M.M.

REDAKTUR PELAKSANA:
Indra Yatini Buryadi, S.Kom., M.Kom.

SEKRETARIS:
Al. Agus Subagyo, S.E., M.Si.

LAYOUT dan PRODUKSI:
Dison Libardo, S.E., M.Kom.

SIRKULASI:
Totok Budioko, S.T.

DOKUMENTASI:
Dra. Torsinawati
Sukar

Majalah Ilmiah FORMAT diterbitkan empat bulan sekali oleh
STMIK AKAKOM dengan ISSN 1410 – 9158

ALAMAT REDAKSI:
STMIK AKAKOM
Jl. Raya Janti, *Ring Road* Timur, Yogyakarta 55198
Telepon : 62-0274-486664
Faksimile : 62-0274-486438 E-mail : format@netexecutive.com

Dari Redaksi

Kami panjatkan puji dan syukur atas rahmat dan berkah dari Tuhan Yang Maha Esa hingga kami dapat menyelesaikan dan menerbitkan majalah Format pada nomor pertama, tahun pertama. Pada tahun yang pertama ini kami mencoba untuk memperluas cakupan materi dari berbagai hasil penelitian dan karya ilmiah, namun tetap sesuai dengan misinya.

Dalam edisi ini para pembaca akan melihat topik-topik mengenai *Implementasi Soap Dalam WEB Service, Pengaruh Pengumuman Dividend Terhadap Stock Return, Pengelolaan User Berbasis Foaf RDF/XML Pada Perpustakaan Digital, Arsitektur Basis Data Terdistribusi Pada Sistim Operasi Tersebar, Perangkat Lunak Simulasi Mesin Moore Untuk Enkripsi Data, Pengendalian Komputer Client Dari Server Melalui WEB, Knowledge Discovery Pada Risked Customer"s Bank Menggunakan Decision Tree, Estimasi Ukuran Perangkat Lunak Dengan Menggunakan LOC (Line Of Code), Perbandingan Model Feedforward Network Dan Recurrent Network Dalam Melakukan Prediksi, Pengembangan Sistem Pencetakan Terpusat Di Laboratorium Terpadu,* yang sekira akan sangat menarik untuk diulas.

Harapan kami semoga apa yang kami suguhkan kali ini dapat membawa manfaat bagi peminat, dan menambah referensi pembaca pada bidang-bidang tertentu. Terima kasih diucapkan, atas saran dan masukan yang telah kami terima demi kemajuan majalah ilmiah ini. Saran, ide dan gagasan dari pembaca tetap kami tunggu untuk perbaikan pada penerbitan edisi mendatang di abad millenium ini.

Daftar Isi

ESTIMASI UKURAN PERANGKAT LUNAK DENGAN MENGGUNAKAN *LOC (LINE OF CODE)*

Febri Nova Lenti

ABSTRAK

Estimasi proyek adalah menaksir atau memperkirakan sumberdaya-sumberdaya yang dibutuhkan untuk menyelesaikan suatu proyek. Ada dua langkah utama dalam menentukan lama proyek dan biaya yang dibutuhkan proyek, yaitu pertama dengan mengestimasi ukuran / besar perangkat lunak dan kedua dengan menggunakan ukuran bersama dengan faktor lingkungan lainnya untuk menaksir usaha dan biaya yang dibutuhkannya. *LOC* adalah salah satu cara untuk mengestimasi ukuran perangkat lunak . LOC mengestimasi secara tipikal dengan menghitung semua instruksi-instruksi sumber dan mengabaikan komentar dan *blank*. Menghitung baris kode secara manual akan memakan waktu dan membosankan, maka sebaiknya dibangun suatu peranti penghitung LOC otomatis.

Keywords : baris kode, Estimasi, *KLOC* , *LOC*, mesin karakter

1. PENDAHULUAN

Manajemen proyek perangkat lunak adalah pendekatan sistematik dan teratur untuk memantapkan keberhasilan pengelolaan usaha dengan menggunakan skala waktu tertentu untuk menghasilkan produk perangkat lunak. Memprediksi "ukuran" dari suatu sistem perangkat lunak akan lebih membantu dan menjadikan proyek lebih mudah. Pengukuran yang dilakukan dalam mengestimasi ukuran program selayaknya mudah digunakan di awal proyek dan mudah dibaca ketika pekerjaan sudah selesai. Selanjutnya perbandingan besar produk estimasi awal dengan pengukuran aktual kemudian digunakan untuk menyediakan umpan balik ke estimator agar dapat membuat estimasi yang lebih akurat.

Estimasi adalah menaksir atau memperkirakan sumberdaya-sumberdaya yang dibutuhkan untuk menyelesaikan suatu proyek. Dua langkah utama dalam menentukan lama proyek dan biaya yang dibutuhkan proyek, yaitu pertama dengan mengestimasi ukuran / besar perangkat lunak dan kedua dengan menggunakan

ukuran bersama dengan faktor lingkungan lainnya untuk menaksir usaha dan biaya yang dibutuhkannya.

Beberapa unit pengukuran perangkat lunak yang paling banyak digunakan, meliputi:

- *Lines of Code (LOC)*
- Function point
- Jumlah node pada diagram alir data (DAD)
- Jumlah entitas pada diagram relasi entitas (ER diagram)
- Jumlah kotak proses (PSPEC) atau Kontak kontrol (CSPEC) pada suatu bagan struktur
- Jumlah pernyataan "sebaiknya" versus pernyataan "harus" dalam spesifikasi kebijakan pemerintah
- Jumlah dokumentasi
- Jumlah obyek, atribut, dan layanan pada diagram obyek

Karakteristik Estimasi

- Estimasi didasarkan pada model probabilitas, bukan deterministik
- Estimasi pada suatu angka dalam suatu selang dari suatu kuantitas yang diestimasi
- Kepercayaan pada estimasi tergantung pada ukuran selang dari kuantitas yang diestimasi
- Kepercayaan pada estimasi tergantung pada ukuran selang dari kuantitas yang diukur
- Informasi yang cukup banyak (sebagai dasar estimasi) akan memperkecil
- Informasi yang sangat relevan yang dipakai sebagai dasar estimasi adalah dalam selang yang kecil

Beberapa kemungkinan penyebab lemahnya estimasi:

- Kita tidak mengembangkan kepakaran estimasi
- Kita tidak membuat cukup syarat untuk menggantikan akibat dari bias
- Kita tidak memiliki cukup pengertian dari apa yang seharusnya diestimasi
- Kita tidak mendasarkan estimasi pada ukuran performansi waktu yang lalu

2. LANDASAN TEORI

[3] LOC mengestimasi secara tipikal dengan menghitung semua instruksi-instruksi sumber dan mengabaikan komentar dan *blank*. sebagai contoh sebuah *tool* otomatis didesain untuk menghitung semicolon (misal pada pascal satu baris ditandai dengan semicolon). Bagaimanapun juga sangat sulit untuk memperkirakan baris-baris kode dari statemen-statemen kebutuhan level tinggi. Pengukuran ini memfasilitasi suatu proses pembelajaran sehingga meningkatkan estimasi. LOC juga memfasilitasi penyimpanan dan pengambilan data ukuran yang dibutuhkan untuk meningkatkan akurasi estimasi. Keunggulan utama dari LOC adalah bahwa LOC berhubungan secara langsung kepada produk yang dibangun. Dengan demikian pengukuran dilakukan setelah adanya fakta yang dibandingkan dengan rencana awal.

[2] Ketika dekomposisi WBS telah sampai pada tingkatan terendah, suatu ukuran " statistik" dapat dibuat melalui suatu pengukuran proses. Ukuran dari tiap komponen dapat diperoleh dengan menanyakan pada tenaga ahli yang sudah pernah mengembangkan sistem yang sama, atau dengan meminta pada pengembang sistem yang berpotensi untuk mengestimasi ukuran dari tiap task pada tingkat terendah dari WBS itu. Ketika ukuran dijumlahkan, jumlah total disebut sebagai penaksiran ukuran " *bottom-up*". Suatu penaksiran yang lebih baik biasanya diperoleh jika masing-masing penaksir diminta untuk menyediakan dalam suatu bentuk taksiran terhadap ukuran optimis, pesimistis, dan realistis. Kemudian dapat dibentuk sebuah distribusi beta dengan mengalikan taksiran ukuran realistis dengan 4, ditambah ukuran optimis dan pesimistis, dan totalnya dibagi 6. Hasil rata-rata ini adalah suatu konversi yang tidak dapat dipisahkan dari ketidakpastian penaksiran. Sebagai contoh, diberikan suatu obyek window yang didekati dengan WBS sistem, kode pendukung yang dibutuhkan untuk proses mengedit window itu diperkirakan antara 200-400 baris kode, dengan nilai keyakinan mendekati 250. Dengan pendekatan nilai optimis dan pesimistis akan menghasilkan perkiraan hasil akhir sebagai berikut:

$$\text{Jumlah baris} = \frac{\text{nilai optimis} + (\text{nilai realistis} \times 4) + \text{nilai pesimis}}{6}$$

$$266 = \frac{200 + (250 \times 4) + 400}{6}$$

Jumlah seribu baris kode disebut sebagai KLOC (Kilo Lines of Code).

Menghitung baris kode secara manual akan memakan waktu dan membosankan, maka kebanyakan organisasi membeli atau membangun suatu peranti penghitung LOC otomatis. Peranti ini dapat menjawab beberapa masalah yang kompleks mengenai pertanyaan tentang persisnya apa yang disebut satu baris kode. Lagipula, tidak jadi soal seperti apa kita menggambarkan LOC, sepanjang definisi tersebut digunakan secara konsisten [1]. Petunjuk perhitungan berikut ini telah digunakan selama bertahun-tahun, untuk merekam ukuran program ada dan untuk menaksir ukuran program yang dikembangkan:

- Pastikan bahwa tiap-tiap " baris kode program" yang terhitung berisi hanya satu statemen program (jika dua statemen yang executable berada pada satu baris, yang dipisahkan oleh suatu titik koma, maka dihitung menjadi dua; jika satu statemen yang *executable* ada atau tersebar ke dua / beberapa baris, maka dihitung satu. Bahasa pemrograman menyediakan pilihan pengkodean untuk bermacam-maccam keperluan, tetapi biasanya tetap mudah untuk menentukan satu statemen tunggal yang *executable* karena *compiler* atau *interpreter* melakukan hal yang sama.
- Hitung semua setoran , termasuk statemen statemen yang executable - pemakai akhir tidak akan secara langsung menggunakan setiap statemen, tapi suatu produk bisa memerlukannya sebagai pendukungnya (seperti utilitas)
- Menghitung definisi data sekali saja
- Jangan menghitung baris yang mengandung komentar
- jangan menghitung kode *debug* atau kode yang bersifat temporer lainnya seperti perangkat lunak pengujian, kasus uji, peranti bantu pengembangan, peranti bantu prototipe, dan seterusnya
- Hitung tiap-tiap defenisi permintaaan, pemanggilan, atau pemasukan (kadang-kadang disebut *compiler directive*) dari suatu makro sebagai bagian dari program yang memakainya (jangan menghitung statemen program yang dipakai ulang)
- Terjemahkan banyaknya baris dari kode program ke baris bahasa asembler padanannya sedemikian sehingga mungkin dapat dibuat perbandingannya terhadap proyek-proyek lain

Kolom pertama dan kedua tabel 2.1. memberikan suatu metoda yang digunakan secara umum untuk menterjemahkan baris kode program / sistem (SLOC) dalam berbagai bahasa ke jumlah rata-rata LOC program asembler. (catat bahwa SLOC

dan LOC dapat saling dipertukarkan) Banyak manager proyek yang menginginkan suatu konversi semua bahasa ke dalam dasar asembler sedemikian sehingga suatu perbandingan satu sama lain bisa dibuat terhadap proyek-proyek. Fungsi lain dari data ini adalah untuk suatu proyek dalam suatu bahasa tertentu akan dikonversi ke bahasa lain. Sebagai contoh, diberikan sautu sistem dengan 50.000 LOC yang ditulis dalam bahasa Cakan dikonversi ke bahasa C++. Dengan menggunakan tabel 2.1., dasar assembler SLOC untuk bahasa C adalah 2,5, maka 50.000 SLOC sistem yang ditulis dalam bahasa C ekivalen dengan 50.000 x 2,5 = 125.000 SLOC jika ditulis dalam Basic Assembler. Dari 125.000 SLOC ini jika ditulis dalam bahasa C++ menjadi 125.000/6 = 20.833 SLOC.

Tabel 2.1. Konversi Bahasa Pemrograman ke Baris Kode Program Basic Assembler per *function point*

Bahasa	Basic Assembler SLOC (Level)	Rerata SLOC per Function Point
Basic Assembler	1	320
Autocoder	1	320
Macro Assembler	1.5	213
C	2.5	128 – 150
DOS Batch Files	2.5	128
Basic	3	107
LOTUS Macro	3	107
ALGOL	3	105 -106
COBOL	3	105 – 107
FORTRAN	3	105 – 106
JOVIAL	3	105 – 107
Mixed Languages(default)	3	105
Pascal	3.5	91
COBOL (ANSI 85)	3.5	91
RPG	4	80
MODULA-2	4.5	80
PL/ I	4.5	80
Concurrent PASC AL	4	80
FORTRAN 95	4.5	71
BASIC (ANSI)	5	64

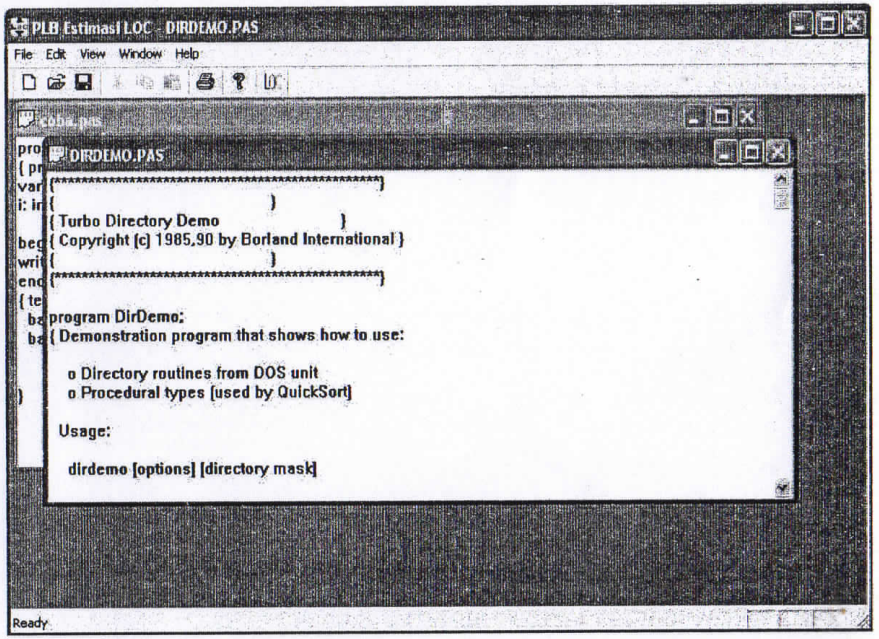
FORTH	5	64
LISP	5	64
PROLOG	5	64
LOGO	5.5	58
Ext Common LISP	5.75	56
RPG III	5.75	56
C++	6	53
JAVA	6	53
YACC	6	53
Ada 95	6.5	49
CICS	7	46
SIMULA	7	46
Database Language	8	40
CLIPPER DB dan Dbase I II	8	40
INFORMIX	8	40
ORACLE dan SYBASE	8	40
Access	8.5	38
Dbase IV	9	36
FileMaker Pro	9	36
Decision Support Languages	9	35
FOXPRO 2.5	9.5	34
APL	10	32
SAS	10	32
DELPHI	11	29
Object-oriented default	11	29
OBJECTIVE-C	12	27
Oracle Developer / 2000	14	23
SMALLTALK	15	21
Awk	15	21
EIFFEL	15	21
UNIX shell Script (PERL)	15	21
4th Generation Default	16	20
Aplication Builder	16	20
COBRA	16	20
Crystal Reports	16	20
Datatrieve	16	20

CLIPPER	17	19
SQL	25	13 – 16
HTML 3.0	22	15
IEF / IEW	23	14
EASYTRIEVE+	25	13
SQL (ANSI)	25	13
EXCELL	50	6
QUATRO PRO	51	6
Graphic Icon Languages	75	4

3. PEMBAHASAN

[1] Dengan membangun suatu perangkat lunak Bantu PLBESTILOC maka dapat dilakukan pengukuran besar program diatas dengan otomatisasi. Untuk tahap tahap yang dilakukan adalah sebagai berikut :

Buka file 'DIRDEMO.PAS' seperti gambar 3.1 pada menu utama PLBESTILOC sebagai berikut:



Gambar 3.1. Membuka file DIRDEMO.PAS

Kode lengkap dari file DIRDEMO.PAS adalah sebagai berikut:

```
{*********************************************}
{                                             }
{ Turbo Directory Demo                        }
{ Copyright (c) 1985,90 by Borland International }
{                                             }
{*********************************************}

program DirDemo;
{ Demonstration program that shows how to use:

    o Directory routines from DOS unit
    o Procedural types (used by QuickSort)

  Usage:

    dirdemo [options] [directory mask]

  Options:

    -W      Wide display
    -N      Sort by file name
    -S      Sort by file size
    -T      Sort by file date and time

  Directory mask:

    Path, Filename, wildcards, etc.

}

{$I-,S-}
{$M 8192,8192,655360}

uses Dos;
```



```
const
  MaxDirSize = 512;
  MonthStr: array[1..12] of string[3] = (
    'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
    'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec');

type
  DirPtr   = ^DirRec;
  DirRec   = record
               Attr: Byte;
               Time: Longint;
               Size: Longint;
               Name: string[12];
             end;
  DirList  = array[0..MaxDirSize - 1] of DirPtr;
  LessFunc = function(X, Y: DirPtr): Boolean;

var
  WideDir: Boolean;
  Count: Integer;
  Less: LessFunc;
  Path: PathStr;
  Dir: DirList;

function NumStr(N, D: Integer): String;
begin
  NumStr[0] := Chr(D);
  while D > 0 do
  begin
    NumStr[D] := Chr(N mod 10 + Ord('0'));
    N := N div 10;
    Dec(D);
  end;
end;

{$F+}

function LessName(X, Y: DirPtr): Boolean;
begin
  LessName := X^.Name < Y^.Name;
end;
```

```
function LessSize(X, Y: DirPtr): Boolean;
begin
  LessSize := X^.Size < Y^.Size;
end;

function LessTime(X, Y: DirPtr): Boolean;
begin
  LessTime := X^.Time > Y^.Time;
end;

{$F-}

procedure QuickSort(L, R: Integer);
var
  I, J: Integer;
  X, Y: DirPtr;
begin
  I := L;
  J := R;
  X := Dir[(L + R) div 2];
  repeat
    while Less(Dir[I], X) do Inc(I);
    while Less(X, Dir[J]) do Dec(J);
    if I <= J then
    begin
      Y := Dir[I];
      Dir[I] := Dir[J];
      Dir[J] := Y;
      Inc(I);
      Dec(J);
    end;
  until I > J;
  if L < J then QuickSort(L, J);
  if I < R then QuickSort(I, R);
end;
```



```
procedure GetCommand;
var
  I,J: Integer;
  Attr: Word;
  S: PathStr;
  D: DirStr;
  N: NameStr;
  E: ExtStr;
  F: File;
begin
  WideDir := False;
  @Less := nil;
  Path := '';
  for I := 1 to ParamCount do
  begin
    S := ParamStr(I);
    if S[1] = '-' then
      for J := 2 to Length(S) do
        case UpCase(S[J]) of
          'N': Less := LessName;
          'S': Less := LessSize;
          'T': Less := LessTime;
          'W': WideDir := True;
        else
          WriteLn('Invalid option: ', S[J]);
          Halt(1);
        end
    else
      Path := S;
  end;
  Path := FExpand(Path);
  if Path[Length(Path)] <> '\' then
  begin
    Assign(F, Path);
    GetFAttr(F, Attr);
    if (DosError = 0) and (Attr and Directory <> 0) then
      Path := Path + '\';
  end;
  FSplit(Path, D, N, E);
  if N = '' then N := '*';
```

```
    if E = '' then E := '.*';
    Path := D + N + E;
  end;

  procedure FindFiles;
  var
    F: SearchRec;
  begin
    Count := 0;
    FindFirst(Path, ReadOnly + Directory + Archive, F);
    while (DosError = 0) and (Count < MaxDirSize) do
    begin
      GetMem(Dir[Count], Length(F.Name) + 10);
      Move(F.Attr, Dir[Count]^, Length(F.Name) + 10);
      Inc(Count);
      FindNext(F);
    end;
  end;

  procedure SortFiles;
  begin
    if (Count <> 0) and (@Less <> nil) then
      QuickSort(0, Count - 1);
  end;

  procedure PrintFiles;
  var
    I, P: Integer;
    Total: Longint;
    T: DateTime;
    N: NameStr;
    E: ExtStr;
  begin
    WriteLn('Directory of ', Path);
    if Count = 0 then
    begin
      WriteLn('No matching files');
      Exit;
    end;
```

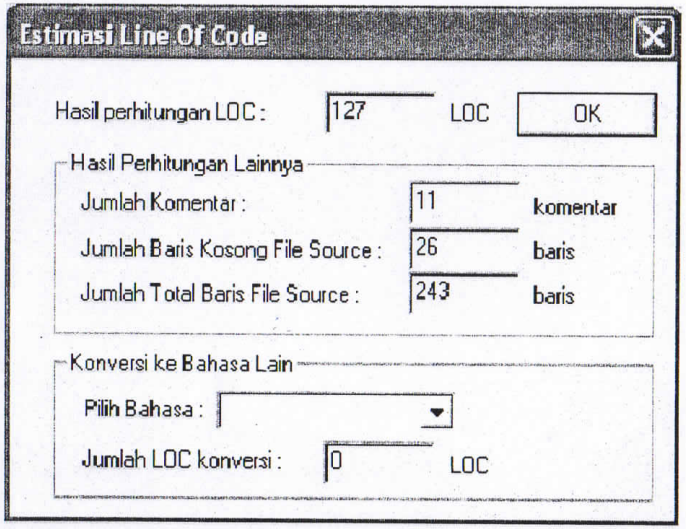


```
  Total := 0;
    for I := 0 to Count-1 do
    with Dir[I]^ do
    begin
      P := Pos('.', Name);
      if P > 1 then
      begin
        N := Copy(Name, 1, P - 1);
        E := Copy(Name, P + 1, 3);
      end else
      begin
        N := Name;
        E := '';
      end;
      Write(N, ' ': 9 - Length(N), E, ' ': 4 - Length(E));
      if WideDir then
      begin
        if Attr and Directory <> 0 then
          Write(' DIR')
        else
          Write((Size + 1023) shr 10: 3, 'k');
        if I and 3 <> 3 then
          Write(' ': 3)
        else
          WriteLn;
      end else
      begin
        if Attr and Directory <> 0 then
          Write('<DIR>    ')
        else
          Write(Size: 8);
        UnpackTime(Time, T);
        WriteLn(T.Day: 4, '-',
          MonthStr[T.Month], '-',
          NumStr(T.Year mod 100, 2),
          T.Hour: 4, ':',
          NumStr(T.Min, 2));
      end;
      Inc(Total, Size);
```

```
    end;
    if WideDir and (Count and 3 <> 0) then WriteLn;
    WriteLn(Count, ' files, ', Total, ' bytes, ',
      DiskFree(Ord(Path[1])-64), ' bytes free');
end;

begin
  GetCommand;
  FindFiles;
  SortFiles;
  PrintFiles;
end.
```

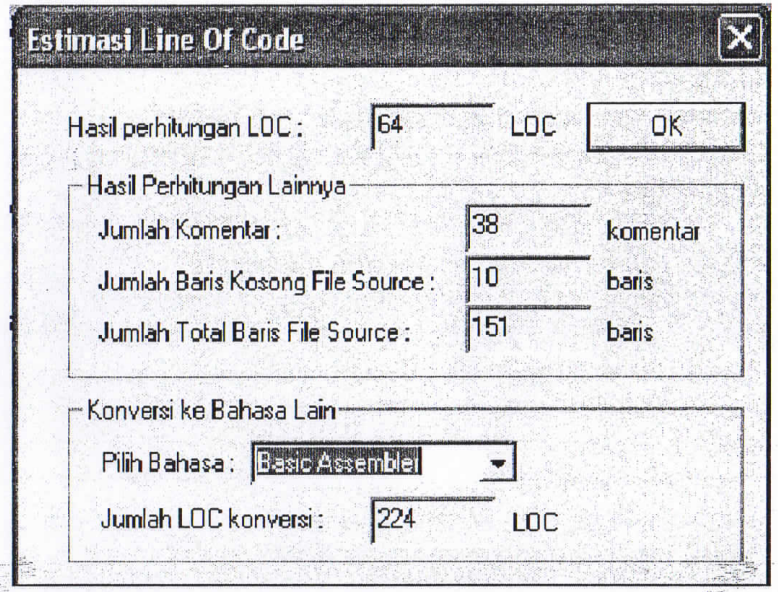
a) Lakukan proses penghitungan LOC
Pilih menu View->Line Of Code pada menu utama PLBESTILOC maka akan muncul jendela berikut (gambar 3.2]:



Gambar.3.2. Jendela 'Estimasi Line Of Code'untuk file 'DIRDEMO.PAS'

b) Konversi LOC ke Bahasa lain
Klik tanda panah bawah pada combo box 'Pilih Bahasa', maka akan muncul daftar bahasa pemrograman pada dropdown combo box, pilih 'Basic Assem

bler', maka pada edit box 'Jumlah LOC konversi' terdapat nilai hasil konversi LOC dari bahasa pascal ke bahasa basic assembler [gambar 3.3].



Gambar 3.3. Hasil konversi LOC 'DIRDEMO.PAS' ke bahasa basic assembler

4. PENUTUP

KELEBIHAN LOC

Kelebihan-kelebihan menggunakan baris kode (LOC) ini sebagai unit pengukuran perangkat lunak meliputi:

- LOC ini sudah umum digunakan dan dapat diterima secara universal
- LOC ini mengijinkan adanya perbandingan matriks ukuran dan produktifitas antara kelompok-kelompok pengembangan yang berbeda beda
- LOC ini berhubungan secara langsung dengan produk akhir
- LOC lebih mudah diukur terhadap penyelesaian proyek
- LOC mengukur perangkat lunak dari sudut pandang pengembang - yaitu apa yang ia kerjakan.
- Teknik estimasi ini memungkinkan adanya aktifitas untuk kenaikan berkelanjutan - ukuran perkiraan dapat dengan mudah dibandingkan dengan ukuran riil ketika

sesudahnya dilakukan analisa terhadap proyek tadi (seperti bagaimana keakuratan perkiraan, mengapa ia bisa mengubah sekian persen?, serta apa yang bisa dipelajari dari sini untuk menilai ukuran proyek berikutnya?

KELEMAHAN LOC

Kelemahan-kelemahan menggunakan baris kode (LOC) adalah sebagai berikut:

- LOC kesulitan untuk menaksir perangkat lunak baru diawal tahap siklus hidup pengembangan
- Instruksi program berbeda menurut jenis bahasa pemrograman, metoda-metoda disain, dan dengan gaya dan kemampuan programmer
- Tidak adanya organisasi standarisasi (seperti ISO) untuk menghitung baris kode
- Perangkat lunak melibatkan banyak biaya-biaya yang mungkin tidak dipertimbangkan ketika melakukan pengukuran kode untuk menaksir biaya – terdapat " biaya-biaya tetap" seperti biaya kebutuhan spesifikasi dan dokumen-dokumen pemakai yag tidak dimasukkan dalam pengkodean.
- Pemrogram mungkin akan menerima bayaran lebih untuk jumlah baris kode yang besar jika pihak manajemen salah tanggap terhadap produktifitas mereka dan tidak sebaliknya untuk pihak desainer, padahal kode program bukanlah yang paling esensi dari suatu produk melainkan pencapaian fungsionalitas dan performansi.
- Penghitungan LOC seharusnya membedakan antara kode yang dihasilkan dengan kode berdasarkan fungsinya - karena ini lebih sulit yaitu menghitung utilitas, bukan hanya sekedar "menghitung langsung" kode, yang sebenarnya bisa diperoleh dari daftar kode compiler
- LOC tidak dapat digunakan untuk normalisasi jika platform atau bahasa dibedakan

DAFTAR PUSTAKA

[1] Febri Nova Lenti, *Laporan Penelitian Perangkat Lunak Bantu Estimasi Besar Perangkat Lunak dengan Menggunakan LOC (PLBESTILOC)*, STMIK AKAKOM, 2007

[2] Futrel R.T., Shafeer D.F., and Shafer L.I., *Quality Software Project Management*, Prentice Hall, 2002

[3] Humprey W.S., *Managing the Software Process*, Addison-Wesley, 1990

[4] Kernighan B.W., Ritchie D.M., *The C Programming Language*, Prentice Hall, Second Edition, 1988.

[5] Liem I., *Diktat Kuliah IF 223 Algoritma dan Pemrograman*, Jurusan Teknik Informatika ITB, 1999

[6] McLeod G., Smith D., *Managing Information Technology Projects*, University of Cape Town

[7] Pressman R.S., *Software Engineering A Practitioner's Approach*, Mc Graw-Hill 1997.